Abstract: Testing modern VLSI circuits is a complex affair. Industries use a fully automated test setup called ATE (Automatic Test Equipment), which comprises of almost all standard test equipments controlled by a central controlling unit. The major drawback of this scheme is the huge monetary cost of an ATE. Our work aims to provide a cheap alternative to the traditional ATE. An Automated Test System for VLSI circuits is developed converting a simple Desk Top PC to a Virtual ATE.

INTRODUCTION

Advances in Fabrication technology has enabled VLSI engineers to go to sub micron technology allowing them to pack millions of transistors in a single chip. As the number of elements inside a chip started increasing exponentially there was also a sharp rise in the post manufacturing defects/faults in the chip. As the elements and their interconnects started becoming smaller and smaller, the probability of interconnects getting short to ground or power line or getting short with each other etc. also went high. Not only interconnects, the faults in components like shorting of drain and source of MOS transistors also went up. So post manufacture testing of VLSI became an important issue.

Generally testing VLSI circuits require a number of discrete test equipments. Traditionally the practice was to control the equipments manually. The disadvantages were slow speed of testing, manual recording of results and use of a general purpose software like EXCEL to compare the recorded data with those expected (provided by design specs). To over come these drawbacks industries started using a very high hand test equipment called ATE which had all standard discrete test equipments interfaced and controlled by a central control unit. ATE is very large, vendor specific, non-scalable and expensive equipment. Traditional ATE costs several millions of dollars and takes several months to build. Therefore these types of traditional ATEs are not available to every test engineer. Now a days with the availability of computer controlled test equipments and test automation software like Labview, Agilent VEE etc. test engineers can create “Personal ATE” on their desktop. The aim of this work is to automate the testing of VLSI circuits by integrating and synchronizing a number of discrete test equipments using a desktop PC. All the instruments are programmable and are connected to the desktop using GIPB cables. Agilent VEE software is used to program the instruments (as required for the specific test), record output response, analyze and display results on the console.

The paper is organized in three broad sections. The first section deals with introduction to ATE, interconnecting buses like GPIB, and Test automation Software. The second section highlights the use of this “Automated Test System” in testing both Analog and Digital VLSI circuits. The methodology shown for digital VLSI circuits is general and can be adopted for
almost all types of digital VLSI circuits. The third section concludes by providing in brief, the pros and cons of using this methodology for VLSI testing.

**Automatic Test Equipment**

Testing modern VLSI systems require a large number of parameters must be measured. Testing also requires analysis of results; comparison with designed specifications and display in an easily interpretable format. It is time consuming and difficult to perform tests and measurements of complex circuit parameters manually using conventional, discrete pieces of test equipments. To cope up with the complexity of testing, Automatic Test Equipment is used which provide tremendous flexibility, allowing many different types of devices to be tested without changes to the test hardware. Software changes reconfigure this type of tester to accommodate different types of devices. It enables VLSI testing at high speed although at a very high monetary-cost.

**1.2 Virtual Instrumentation**

VI is defined as any type of test instrumentation that is basically software reliant and primarily dependent on a computer to control test hardware and equipment, analyze, and present test results. Software is the key to virtual instruments. The power of VI application software lies in the fact that it empowers the user to include test equipments as objects in their programs.

**1.3 GPIB**

General Purpose Interface Bus (GPIB) is a versatile and reliable bus system especially designed for connecting computers and instruments. Its vendor specific variant is called HPIB (Hewlett Packard Interface Bus). This networked system has all features that are required to create a test system. The GPIB is an IEEE standard bus system and the standard's number is IEEE488.1. Sharing same GPIB Bus system up to 15 instruments, called *devices*, can be connected to one computer, usually called *controller*, because it is responsible for coordinating bus traffic.

**1.3 Test Automation Software**

Test automation Software that controls GPIB bus is available off the shelf or can be developed in-house. National Instrument's Labview and Agilent's (formerly Hewlett-Packard's) Agilent-VEE are two of the most popular software packages for controlling instruments via the GPIB/IEEE-488 bus. Alternatively, control software can also be developed using traditional "C" programming, given a certain level of expertise. However, use of the Labview or HP-VEE allows test engineer to design sophisticated programs quickly and easily. Labview and HP-VEE come with software modules for driving most common instruments.

**2. CASE STUDY**

**2.1 Implementation of Analog Design For Testability on VEE Environment**

A work has been done to implement Analog Design for Testability (ADFT) by pseudo random analog signal to circuit under test analogous to the Signature Analysis in Digital Domain. The circuit chosen for testing is a simple second order filter realized using an opamp, resistances and capacitances. The circuit schematic is shown in fig 2.1. Catastrophic faults were introduced in
the circuit by sorting a high resistance with a small resistance for short circuit faults and putting a high resistance in series for open circuit faults. Fault simulation is done in SPICE by changing the circuit script file, to incorporate the catastrophic faults described above. A random signal applied to the input of both the normal and the faulty circuit. From simulation it has is found that the faulty circuit response to the random sequence is different from that of the fault free circuit. Thus this method will serve as a tool for distinguishing fault free circuits from faulty circuits.

The challenge is to verify the simulation results experimentally. The input random signal is to be a very precise signal with accuracy of $10^{-4}$ volts. Similarly the response of the circuit has to be recorded up to four places of decimal. The difference between the output response of the fault free circuit and the faulty circuit is very small to be easily marred by spurious noise.

2.1.1 Inputting a pseudo random sequence of high accuracy

The input sequence is a periodic repetition of 1000 random samples placed at a distance of 10 microseconds. This signal has to be generated continuously from the arbitrary waveform generator. An Agilent 33250A 80 MHz Function/Arbitrary waveform generators is first loaded with the data pattern from a file by GPIB interface. The pattern generator is automatically programmed to output the data at a desired frequency. The frequency is periodically repeated. Since pattern generator has very high-resolution data storage the quantization error is much smaller than $10^{-4}$ V.

2.1.2 Recording the response of the circuit

An Agilent 54615B oscilloscope (with GPIB programmability) records the response of the circuit. As the oscilloscope samples at the rate 500 Mega Samples/s, it is ensured that all signal points get recorded. Also high resolution of the oscilloscope ensured quantization error insignificant in comparison to accuracy required. The VEE environment records the oscilloscope screen data and stores them in a file in Microsoft XL format. Without Automated Test Setup it is very difficult to carry out this test manually. Automation helps to send thousands of samples generated by a SPICE Pseudo Random Pattern Generator to Waveform generator’s memory. Similarly the data from Oscilloscope is directly measured and stored in an excel format. So storing a huge amount of data no longer remains a bottleneck.

2.1.3 The Arrangement

![Fig 2.1: Setup & Circuit Schematic](image)
Fig 2.2 The VEE Program

Fig 2.3 Experimental Result:
2.2 Structural Testing of Digital VLSI circuits Using Desktop ATS

2.2.1 Device Under Test

This section deals in detail with the concepts of structural testing of Digital VLSI circuits. The digital circuit under consideration is a sixteen bit cellular automata / pattern generator which would be referred to as CA-16 in this paper. CA-16 is a 16 bit Linear Feed Back Shift Register designed in Advanced VLSI Design Laboratory, IIT Kharagpur. The brief behavioral code of the circuit is given in fig 2.4.

always @(posedge CLK)
    begin
        if(control AND control1) c=INP;
    end

always @(posedge CLK)
    begin
        if(!control AND !control1)
            begin
                OUT=e; d=OUT;
            end
    end

always @( d or c)
begin
    e[0] = (d[0] AND c[0]) OR d[1];
    for(i=1; <15; i=i+1) e[i] = d[i-1] OR (d[i] AND c[i]) OR d[i+1];
end

Fig 2.4: Behavioral Pseudocode of CA-16

The input port consists of 16-bit Data Bus and two control lines. The operation is controlled by the input conditions on the control lines. It has three 16-bit registers to carry out the internal operations. Initially the seed vector is fed in as the input by setting the two control lines to logic one. Then to start random pattern generation the control lines are set to logic zero. The output is sent through port “OUT”.

VI based Automatic Test System was used to automate the testing of CA-16. Initially Functional testing was performed to check whether the designed system was free of implementation errors (verification test). The objective of the Functional testing was to validate the operation of the system with respect to its functional specifications. CA-16 has 16-bit input, excluding control lines so totally exhaustive functional testing would require as high as $2^{16}$ different input vectors. To overcome the difficulty of such a huge number of vectors Structural Testing of CA 16 was performed with approximately 100% coverage.

2.2.2 Functional test of Digital Circuits:

The objective of the functional testing is to validate the correct operation of the system with respect to its functional specifications. Functional Testing is carried out for a small number of vector combinations. Initially the circuit is simulated in Verilog by giving stimulus from a test bench. Once the designer is satisfied that the simulation results are as desired the design under
goes some other steps like synthesis, layout and finally sent for fabrication. Once the silicon comes from fabrication laboratory, the next step comprises of manufacturing a test PCB and mounting the chip. The PCB has all required connections made, which enable access of the pins of the chip by different test instruments. After the chip is mounted on a PCB a Pattern Generator is programmed to provide the chip with the same vectors as in the Verilog test bench. The output response is recorded in a Logic Analysis System and compared with Verilog Output response. This consists of translating the Verilog test bench to signals and feeding them as Zeros and Ones to the pattern generator and comparing Logic Analyzer response with simulation response. This type of testing is possible manually only if the numbers of vectors to be tested for are small and data bits are manageable. Functional tests detecting any fault present in the system is called exhaustive test and thus all possible input vectors must be applied to the system. Because, of the length of the resulting tests, exhaustive tests can be applied to only small circuits. Having some knowledge of the structure of the circuit and introducing DFT measures like Scan Chains or BIST almost 100% fault coverage can be obtained with sufficiently less number of test vectors. This type of testing is called structural testing where we need computer-aided tools to generate vectors, ATE to feed patterns automatically to Pattern Generator and Compare Logic Analyzer response to simulation response. To keep monetary cost of testing low VI based Automatic Test System was developed to carry all the above-mentioned tasks.

2.2.3 Design For Testability

As already mentioned bit width of the input data bus is 16. Hence the exhaustive testing would require as high as $2^{16}$ different vector combinations. As the circuit is sequential (data is stored within the design and a clock input is used to drive the circuit) this creates a potentially large problem: to test for a particular fault may require the inputs and the stored data to be in a particular state. To place the stored data into a particular state by normal operation may require many clock cycles and/or input combinations. For example, for a circuit that contains a 20 bit up counter that is initialized to 0 when the circuit is reset. If it is required that the MSB of this counter should be 1 for a particular test then to reach this state, $2^{19}$ (=524288) clock cycles would be required. This problem would clearly lead to an unacceptable overhead associated with this test. What is required is a quick method of setting the state of the stored data anywhere in a circuit without expending a large number of clock cycles.

This problem can be addressed by looking at the structure of normal, sequential circuits:

The circuit appears to be stages of logic separated by stages of data storage in flip-flops. All the flip-flops are driven by the same clock (in practice, the clock is arranged as a tree of buffers so that each flip-flop is clocked at nearly the same instant) and, in this way, the behavior of the circuit is reliable and deterministic. However, if the flip-flops could be loaded independently during testing then they could be set to any arbitrary state before each test allowing, in effect, the problem of testing a sequential circuit to be changed to testing a combinatorial circuit with prior initialization. To do this, the flip-flops have been replaced by scan flip-flops:
If SE (Scan Enable) is 0, then the input to the flip-flop is derived from the normal D input. However, if SE is 1 (as it would be during test), then the input to the flip-flop comes from the Serial Input (SI) and this is used to set up the state of the flip-flops before each test.

The output of a flip-flop is connected to the SI of another flip-flop to form a long shift-register - a scan chain. Data can be serially loaded into the chain of flip-flops to set their state to the input values needed to test each block of combinatorial logic and then, with Test disabled, the circuit is clocked once which stores the output of the combinatorial logic into the flip-flops. Test is enabled again and the output values can be unloaded (for comparison with their expected values) using the scan chain whilst the next test inputs are loaded in. This sort of scheme is a trade-off: it still takes a number of clock cycles to set up a test (equal to the number of flip-flops in a scan chain) but the external overhead is very low: only 3 pins are required - Test, Scan-In, and Scan-Out. It is clear that scan testing is compatible with JTAG and JTAG can be used as the access mechanism at the PCB and system level to load and unload the scan chains with individual ICs.

Unfortunately, a typical design can contain a lot of flip-flops and it may be impractical to connect them all together. The scan-path can be very long and take a long time to load, and it will not be possible to partition the design and test functional parts separately. To overcome this problem, a design usually contains multiple, shorter scan-paths associated with separate functional parts. This does complicate the interface (there will be a Scan-In and Scan-Out pin for each scan path or these pins can be multiplexed) but can improve the utility of the method.

2.2.4 DFT Implemented in CA-16:

All Flip Flops are multiplexed Scan FFs. There are in all 48 FFs in the three 16 bit registers, which are connected in 3 scan chains each, having 16 FFs. The scan chain List is shown in fig 2.5.

Complete scan chain #1 (test_si[2] --> test_so1a) contains 16 cells:

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b_reg[0]</td>
<td>b_reg[9]</td>
</tr>
<tr>
<td>b_reg[1]</td>
<td>b_reg[10]</td>
</tr>
<tr>
<td>b_reg[7]</td>
<td>test_so1a</td>
</tr>
</tbody>
</table>
Complete scan chain #2 (test_si[1] --> test_so2a) contains 16 cells:


Complete scan chain #3 (test_si[0] --> test_so3a) contains 16 cells:


Fig2.5: Scan Chain implemented in CA-16

This DFT measure adds four input lines to the already existing ones. Three constitutes the lines through which the scan data is feed in; test_si[0], test_si[1], test_si[1]. The fourth line is the scan enable input. Three lines are also added to the output to scan out the data from the three scan chains.

2.2.5 Structural Test Procedure:

The first step is to use any ATPG (Automatic Test Pattern Generator) tool “Design Analyzer” to generate the Test Patterns. The ATPG CAD tool takes the net-list of the circuit with DFT (Scan Chain) implemented as input and automatically generates the test program in Verilog format. It also generates the expected primary and scan out response for the fault free circuit. The test program consists of all the vectors required to detect the observable faults. As stated already, to accomplish these the Pattern Generator needs to be configured by typing in the vectors as series of zeroes and ones; comparing Logic Analyzer response to the expected response. As the number of vectors rise steeply for larger circuits these steps are too complex to be carried out manually. Hence an Automatic Test System was developed in VEE to accomplish all these. In brief the sequence of steps are as follows:
1. Execute Synopsys ATPG software to generate Test Vectors: File Format - Verilog. Fig 2.6 shows the ATP file with the First Test Vector.

**Fig 2.6 ATP File generated by Synopsys “Design Analyzer”**

`timescale 1ns / 10ps

module TOP_ctl ;

integer _failed ;
reg [0:0] _ck;
reg [2:0] _si;
reg [18:0] _pi;
reg [2:0] _e_so, _m_so;
wire [2:0] _r_so;
event _check_so ;
reg [15:0] _e_po, _m_po;
wire [15:0] _r_po;
event _check_po ;

/* Synopsys Test Compiler, 2000.05-1 (Jul 12, 2000) was used to generate this pattern set */
/* INPUT VECTOR FILE = test_pattern.vdb was the source file for this pattern set */

assign _ck=1'b0; /*clock*/
assign _pi=19'bXXXXXXXXXXXXXXXXXXXXX1; 
assign _si=3'bXXX;

/* Pattern 1 */

#205 ; /* delay 205 */
assign _pi=19'bXXXXXXXXXXXXXXXXXXXX111;............................(A)

/* Primary input consists of 16 bit input data, two control inputs; and the scan enable line */

assign _si=3'b011; /* Scan input */ .......(B)
The Verilog ATP file needs some explanation. The primary input is 19-bit, named _pi. It includes 16-bit input data, two control lines, and the scan enable line. It is to be observed that initially the scan enable bit (marked as Point A) is high (it is the LSB bit of _pi). The next step is to set the FFs as required. This is accomplished by feeding in required data values in the three scan lines. The three scan lines are indicated by a 3-bit port named _si (Shown as point B in the file). Once the Scan Chains are set as...
required, the output response on the scan out port is to be saved in a Logic Analysis System. This event is indicated by \_check\_so event (shown as point D). The expected scan out data for the fault free circuit is shown by variable \_e\_so (shown as point C). After the Scan Chains are set the scan enable line is disabled (shown as point E) and a vector is given as normal input. A single clock pulse is applied and the data at the primary output port is to be recorded in Logic Analyzer. Similarly, in the case of Primary output there is a need to compare the expected primary output (shown as point F) with recorded output.

2. The vectors in the Verilog file are to be fed to the Pattern Generator. The Pattern generator cannot take Verilog format as input. The pattern Generator requires ASCII format, which is shown in fig 2.7.

```
ASCII

/*The first line needs to by typed as ASCII */

ASCDOWN

FORM: MODE FULL

/*Full channel Mode uses all the available points of a pattern generator card */

LABEL VECT, 24 /* number of signals needed to be fed */

VECT

M* /*Vectors start */

/* main start */

................

1E 34 EA /* vectors in Hex */

5E AC B0

Fig 2.7. ASCII file format required by Pattern Generator
```

The pattern generator feeds the vectors typed (in Hex. Format) in the VECT list at a specified frequency. In the ASCII File apart from the vectors few more instructions need to be provided. The most important being the number of channels required. Here 24 channels are marked for use; 16 for input data, 2 for Control lines, 1 for system clock, 1 for scan enable, 3 for scan input, and 1 for record event. The record event signal is generated when there is a requirement to store the primary output or scan output in Logic Analyzer (\_check\_so or \_ckeck\_po).

Software in VEE has been written which configures the Pattern Generator and converts the Test Vector file format from Verilog to ASCII. The Software also optimizes the required number of data channels, which are required by pattern generator to give input stimulus to the chip. The software analyzes the pattern required by any channel throughout the test cycle. If any channel is in “don’t care” state throughout the cycle it
can be hard wired to Gnd or VDD. Similarly, if two or more channels need to generate the same pattern throughout the test cycle, those can be merged. If any channel has a constant value throughout then that point can be hard wired as required.

3. Logic Analyzer configuration requires the number of channels to be probed (in this case 16 primary output and 3 scan output) and the value of logic threshold to be set for those channels. The sampling frequency is set more than or equal to twice the frequency of the operating circuit.

4. Test setup for the chip, requires mounting the CHIP in a test board (a PCB is designed to mount the chip) and connecting the pins to Logic Analyzer/Pattern Generator probe points.

5. Save the output response from scan out and primary output pins of the chip in Logic Analyzer.

6. Automatic comparisons of results that are recorded by Logic Analyzer with that of primary output and scan output response expected by the SYNOPYS ATPG -- “Design Analyzer”.

7. Automatic determination of the test coverage.

The software written in VEE automates all these steps. So the job of the test engineer comes down to just feeding in the circuit netlist to VEE software; mounting the chip on test PCB and making physical connections (to Pattern Generator and Logic Analyzer). Rest all things are taken care of by the Automatic Test System designed. In this system Agilent 16700 series Logic Analysis System with built in Pattern Generator cards was used. Synopsys “Design Analyzer“ was used for automatic pattern Generation.

3.Conclusion:

Automatic Test system developed by this methodology gives a cheap alternative to the traditional ATE. The test engineer feels that all the instruments are software objects, which can be included in his Object Oriented Software. This level of abstraction helps the engineer to develop software with ease to control all required instruments from a central computing unit. This system suffers from some drawbacks, which mainly include the delay in transmission of data & control information from PC to the instrument by the GPIB cables. So use of very long cables is not recommended. The solution is to transfer the whole control information and data to the instrument’s memory before activating it. This can be accomplished if the instrument has an on device user memory. By use of fast Data Acquisition Cards this transfer delay can be minimized. In all it can be stated that this approach is a very cheap alternative to ATE.
Acknowledgement:
The authors are thankful to Dr S. Mukhopadhyay, Asst. Professor, Dept. of Electrical Engineering, IIT Kharagpur, for his immense help throughout the work.

REFERENCES
5. “SOLD” –Synopsys documentation on Design Analyzer.